**MULTIPLE INHERITANCE: VIRTUAL CLASS**

1. Why virtual?

Create a class base with the public member function print() that simply displays “I am an

Instance of class base.”. From base derive the classes sub1 and sub2 and override their

respective print() function accordingly. Then write a global function who\_are\_you(base &b) that simply calls b.print() and test this function with instances of all three classes. What happens?

1. Virtual functions

Change the definition of base::print() in the previous example to virtual. How does the behavior of the code change? Furthermore, derive a third subclass sub3 from base, for which you do not override the function print(). Test the behavior of instances of this class as well.

1. Pure virtual functions

Now make the definition of base::print() pure virtual. The syntax for that is as follows.

class base {

public:

virtual void print() = 0;

};

Try to compile the code without any further changes. The compiler should complain that you cannot allocate an object of type sub3 since the virtual function base::print() is abstract. Remedy the problem by giving an appropriate definition of sub3::print(). Compile and run the code again.

* 1. Overloading vs. overriding

Create a simple base class with a virtual method print(), that prints something like "base".

Derive a sub class from base that overrides the print() method. Use a base class pointer in the

main program to point to an object of the sub class and call the print() method with the help of

that pointer. You should see the output from the overridden version.

Change the definition of print() in the sub class to print(int i = 0). What happens now in

your (unchanged) main program?

Note: Although you have provided a default print() version in your derived class, C++ does not

consider this to be an overriding definition to the inherited virtual function print().

* 1. Inheriting the virtual nature of a function

Create the obvious hierarchy of classes base → sub → subsub. Provide a virtual method for the

base class. Override that method in both derived classes. Now use a sub class pointer in the main program to point to an object of the subsub class and call the overridden method with the help of that pointer. Which version is executed and why?